Smart Contract Security Audit For:
Racing Raptors

Date: January 1, 2023
Pre-deployment Audit
Authored By: Dev E. Loper
Language: Solidity

Disclaimer

This smart contract audit may reveal client information.

This information may include, but is not limited to:

- Confidential client information
- Proprietary client information
- System vulnerabilities
- Intellectual property
- Methods to exploit system vulnerabilities

The security and/or dissemination of this report subsequent to MacNerd delivering to the client is at the sole discretion and responsibility of the client. MacNerd will not be held liable for any of the information contained in this report after delivery to the client.

Project Scope

MacNerd has tested this smart contract for commonly-known vulnerabilities.

These vulnerabilities include, but are not limited to:

- Re-entrancy
- Pseudo-Randomness Generation
- Gas Limit and Loops
- DoS with Unexpected Throw
- DoS with Block Gad Limit
- Loss of Customer Funds
- Unprotected Self-Destructs
- Access Control
- Unfixed Solidity Version
- Control of Flow

Goals

- Analyze the contract to find any security flaws that might result in the loss of customer funds.
- Find opportunities for gas optimization.

Contents

Informational Summary

MacNerd performed a manual and automate assessment. The Solidity smart contract for Racing Raptors is well secured.

Automated checks were performed with Slither, Remix IDE & Hardhat Compiler.

Total Issues	0
Critical	0
High	0
Medium	0
Low	0
Very Low	0
Informational	0

Evaluation and Code Review

MacNerd has examined contract architecture to ensure that contracts standards and best practices are being used.

The Racing Raptors team did not provide the unit test strips.

Functional Testing

Functional testing is performed to ensure that all the services employed in the blockchain application are working as expected.

The functional testing MacNerd performed on Racing Raptors verifies that all services are working as expected.

Documentation

We were given the Racing Raptors source code as a Github Link:

https://github.com/JEflyer/Racing-Raptors-V3/blob/main/contracts/secondaryMinter.sol

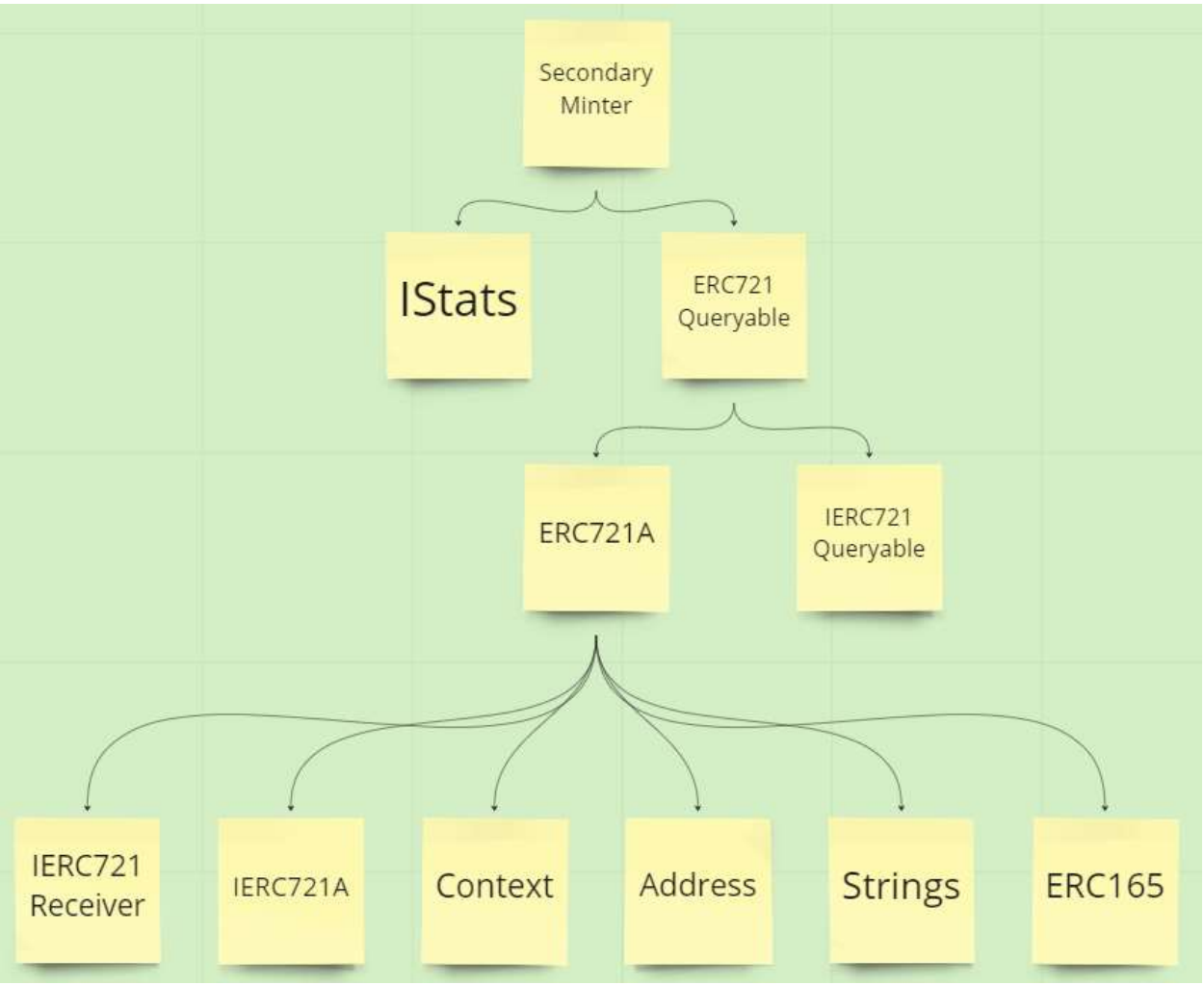
The code was well commented & this is recommended as a best practice.

File & Function Report

Observation: Passed
Test Report: Passed

ID	Function	Type	Observation	Test Report	Conclusion	Score
1	mint	write	Passed	All Passed	No Issue	Passed
2	changeAdmin	write	Passed	All Passed	No Issue	Passed
3	setTokenURI	write	Passed	All Passed	No Issue	Passed
4	tokenURI	read	Passed	All Passed	No Issue	Passed
5	burn	write	Passed	All Passed	No Issue	Passed
6	burnByStats	write	Passed	All Passed	No Issue	Passed

Instance Hierarchy



Severity Definitions

The definitions for risk levels are defined as follows:

Critical: A high-risk vulnerability that can be easily exploited, leading to loss of funds.

High: A high-risk vulnerability that is more difficult to exploit, but can be equally damaging.

Medium: An important vulnerability that needs to be addressed, but rarely leads to loos of funds.

Low: A low-risk vulnerability that usually involved outdated code or unnecessary functions.

Informational: Non-vulnerabilities that are opportunities for code improvement.

Audit Findings

MacNerd found the following vulnerabilities in each of the defined categories listed below.

Critical:
No critical severity vulnerabilities were found.

High:
No high severity vulnerabilities were found.

Medium:
No medium severity vulnerabilities were found.

Low:
No low severity vulnerabilities were found.

Very Low:
No very low severity vulnerabilities were found.

Informational:
No informational points were found.

Summary

Based on the contract provided to MacNerd by Racing Raptors, the security state of the reviewed contract is well-secured.

This is based on MacNerd having used static analyzers as well as manual testing. Zero (0) issues were found.

Created By

This report was created by the security team at https://MacNerd.io. If you have any questions, please reach out to us at contact@macnerd.io